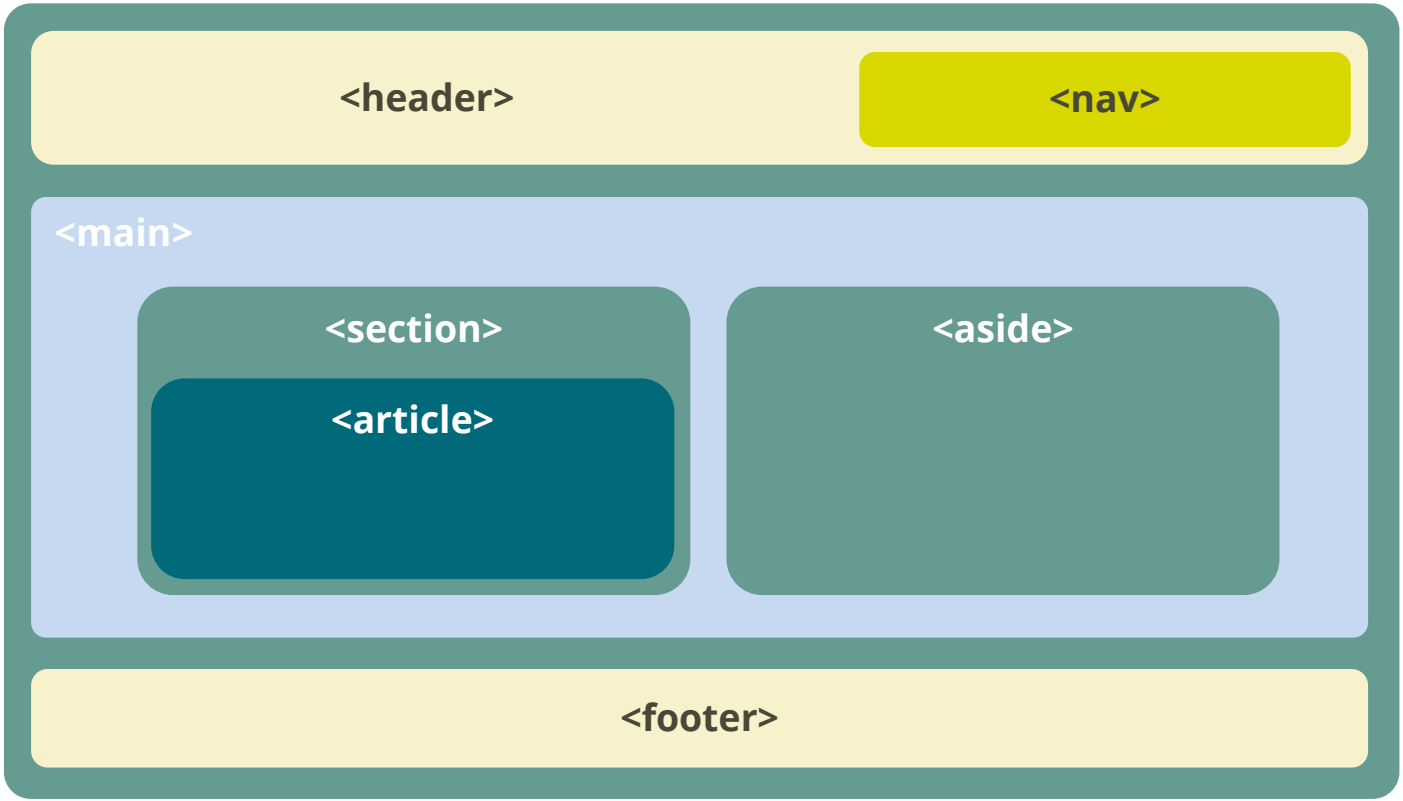
Layout

Introducción al CSS

Introducción al CSS

¿De qué está hecha la web?

Estructura semántica HTML5



Introducción al CSS

- CSS es el acrónimo de **Cascading Style Sheets.**

- El **CSS permite estilar y maquetar páginas web**, modificando parámetros como la fuente, el color o el tamaño del contenido, organizarlo en columnas o añadir animaciones.

- El navegador aplica estos cambios al documento a través de **reglas CSS**. Pero la presentación no tendrá porqué ser igual en todos los navegadores.

- Una regla es un **conjunto de propiedades** con sus correspondientes **valores**, todas ellas agrupadas bajo un **selector**.

- La **hoja de estilos**, por tanto, es el conjunto de **reglas CSS** que afectan a una página web.

Características del CSS

- Permite segregación de responsabilidades
- Separación de presentación y contenido
- Mejora la mantenibilidad
- Se reduce la duplicidad
- Podemos crear versiones para distintos dispositivos

Historia del CSS

Las distintas versiones del CSS se distinguen actualmente en niveles de implementación

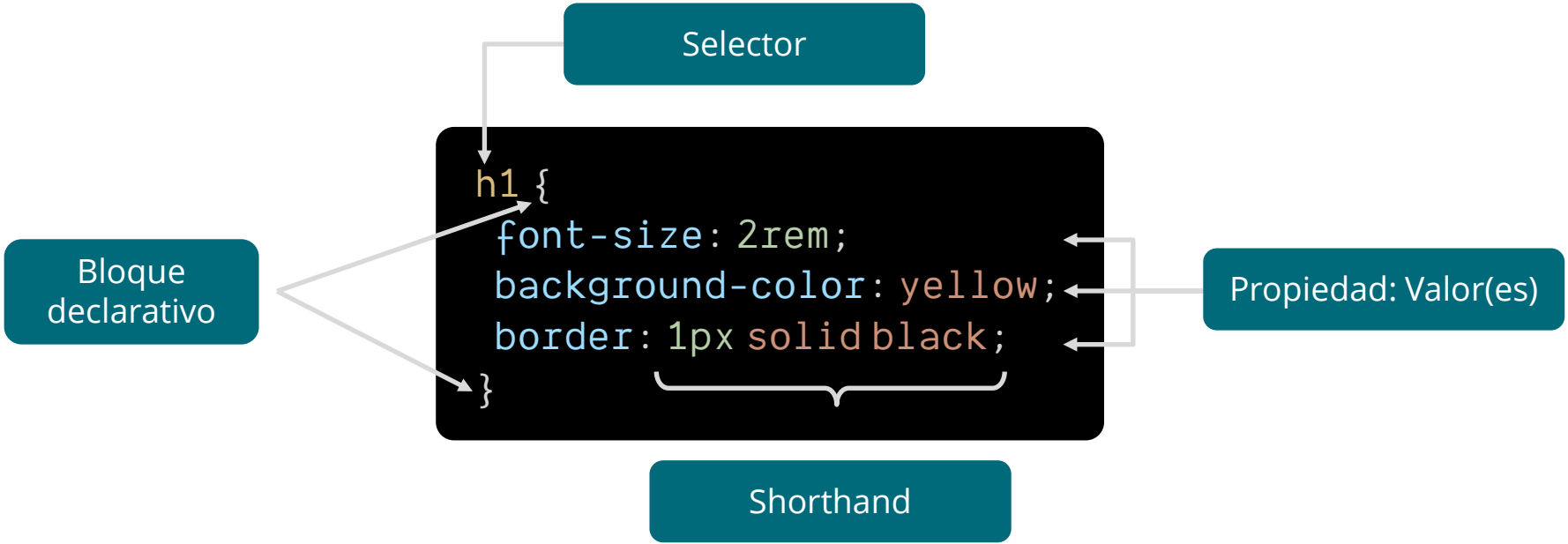
CSS 1 1996	Propiedades de fuente, colores, alineación de elementos, etc.
CSS 2 1998	Propiedades de posicionamiento, tipos de medios, entre otras características
CSS 3 2011	Inicio de módulos separados con funcionalidades nuevas (niveles de versiones)

Sintaxis básica

```
selector { propiedad: valor; }
```

- **Selector:** a qué elemento HTML vamos a aplicar un estilo concreto
- **Propiedad:** la propiedad que vamos manipular, por ejemplo, el margen derecho "margin-right"
- **Valor:** es el valor que vamos a asignar a la propiedad, por ejemplo, para "margin-right" podemos asignar "8px"

Sintaxis básica




```
border-width: 1px;
border-style: solid;
border-color: black;
```

Más info sobre todas las propiedades

Dónde escribir el CSS

CSS Externo

index.html

```
<!DOCTYPE html>
<html>
 <head>
  <link rel="stylesheet"
href="styles.css" />
 </head>
 <body>
  <p>Hello world</p>
 </body>
</html>
```

styles.css

```
p {
 color: red;
}
```

CSS Interno

index.html

```
<!DOCTYPE html>
<html>
 <head>
  <style>
    p {
     color: red;
    }
  </style>
 </head>
 <body>
  <p>Hello world</p>
 </body>
</html>
```

CSS Inline

index.html

```
<!DOCTYPE html>
<html>
 <head>
 </head>
 <body>
  <p style="color: red;">Hello
world</p>
 </body>
</html>
```

Selectores

Es un mecanismo para hacer target sobre elementos HTML

Determinan sobre qué elemento se va a aplicar la regla CSS de la que forman parte

También, en ciertos casos, bajo qué estado se va a aplicar dicha regla CSS

Selectores

Hay una enorme variedad de selectores

```
selector #id .clase :pseudoclase ::pseudoelemento [atributo] {
      propiedad: valor;
}
```

Referencia a selectores en MDN

Selectores simples

```
selector #id .clase :pseudoclase ::pseudoelemento [atributo] {
      propiedad: valor;
}
```

Selectores simples: selecciona elementos **según el tipo de elemento** (tag), **clase** (.class) o **id** (#id)

Selectores simples

```
h1 { }

.miSuperClase { }

#idElemento { }

h1, h2, .title { }
```

EJEMPLO

Selectores de atributo

```
selector #id .clase :pseudoclase ::pseudoelemento [atributo] {
    propiedad: valor;
}
```

Selectores de atributo: selecciona elementos **según el valor** de sus atributos

Selectores de atributo

Patrón	Descripción
[title]	Existe el atributo title
[title="Email"]	Existe el atributo title y es igual a Email
[title~="logo"]	Existe el atributo title y posee una lista de valores que contiene "logo"
[lang|="en"]	Existe el atributo lang y contiene una lista de valores que alguno contiene "en“ o comienza con “en-”
[href^="https"]	Existe el atributo href y comienza con "https"
[href$=".pdf"]	Existe el atributo href y termina con ".pdf"
[href*="lemoncode"]	Existe el atributo href y contiene "lemoncode“ en cualquier posición

Selectores de atributo

```
a[title="Email"] { }

button[selected] { }
```

EJEMPLO

Selectores pseudo clases

```
selector #id .clase :pseudoclase ::pseudoelemento [atributo] {
    propiedad: valor;
}
```

Pseudo clases: selecciona elementos **según su comportamiento o estado** (hover, disabled, etc...)

Selectores pseudo clases

```
a:visited { }

elemento:hover { }
elemento:active { }
elemento:focus { }
```

EJEMPLO

Selectores pseudo elementos

```
selector #id .clase :pseudoclase ::pseudoelemento [atributo] {
    propiedad: valor;
}
```

Pseudo elementos: selecciona elementos **según su situación en relación a otro elemento** (primera palabra del párrafo, contenido que sigue a un elemento, etc.)

Selectores pseudo elementos

```
p::selection { }

p::first-line { }

p::first-letter { }

p::before { }

p::after { }
```

EJEMPLO

Otros selectores

Selectores múltiples: agrupación de **múltiples selectores** separados por comas

Combinaciones: **combinaciones de dos o más selectores** (párrafos situados justo después de títulos, por ejemplo)

Selectores de combinaciones

Selector	Ejemplo	Descripción
Selectores descendientes (descendant selector)	div p	Selecciona todos los p dentro de un div
Selector de hijos (child selector)	h1 > p	Selecciona todos lo p descendientes directos de un h1
Hermano adyacente (adjacent sibling)	p + img	Selecciona los img que sean hermanos inmediatos de un p
Hermano general (general sibling)	h1 ~ p	Selecciona los p que son hermanos de un h1

EJEMPLO

Especificando valores CSS

Números, longitudes y porcentajes

Tipo de datos	Descripción
<integer>	Es un número entero como 1024 o -55
<number>	Representa un número decimal; puede tener o no un punto de separación decimal
<dimension>	Es un <number> con una unidad asociada. <dimension> es una categoría general que incluye los tipos <length>, <angle>, <time> y <resolution>
<percentage>	Representa una fracción de otro valor. Son siempre relativos a otra cantidad

Especificando valores CSS

Unidades absolutas:
- px: píxeles
- mm, cm, Q (cuartos de milímetros), in, pc (picas), pt (puntos)
- s: segundos
- deg: grados

Unidades relativas:
- em: relativo al tamaño de fuente del navegador
- rem: relativo al tamaño de fuente del elemento root
- %: relativo a la herencia
- vw: relativo al ancho del viewport
- vh: relativo al alto del viewport

Colores:
- black, green, ...
- #ff0000 = #f00
- rgb(rojo, verde, azul), rgba, hsl, hsla

Fuentes:
- Roboto, “Times New Roman”, ...
- sans-serif, serif, fantasy, ...

Notación funcional:
- calc
- translate
- rotate
- url

Concepto de cascada

- Un elemento puede verse afectado por más de una regla CSS
- ¿Y si varias reglas CSS colisionan? ¿Cuál se aplica? ¿Cuál gana?
- Cascada = el **orden** de las reglas CSS **importa**
- Pero no solo el orden, es algo más complejo y atiende a 3 factores:

Importancia

En CSS podemos usar una sintaxis específica para asegurarnos que una regla siempre "gana" sobre todas las demás.

```
.textbox {
  background-color: gray;
  border: none !important;
}
```

Especificidad

Establece un sistema de prevalencia de selectores basado en pesos.

Cuanto más específico un selector, más peso y más prevalencia tendrá sobre otros.

Orden de aparición

Si coinciden en especificidad, el orden del código decide.

```
p {
  color: blue;
}

p {
  color: red;
}
```

Concepto de cascada

IMPORTANCIA

{ **propiedad: valor !important** }

Concepto de cascada

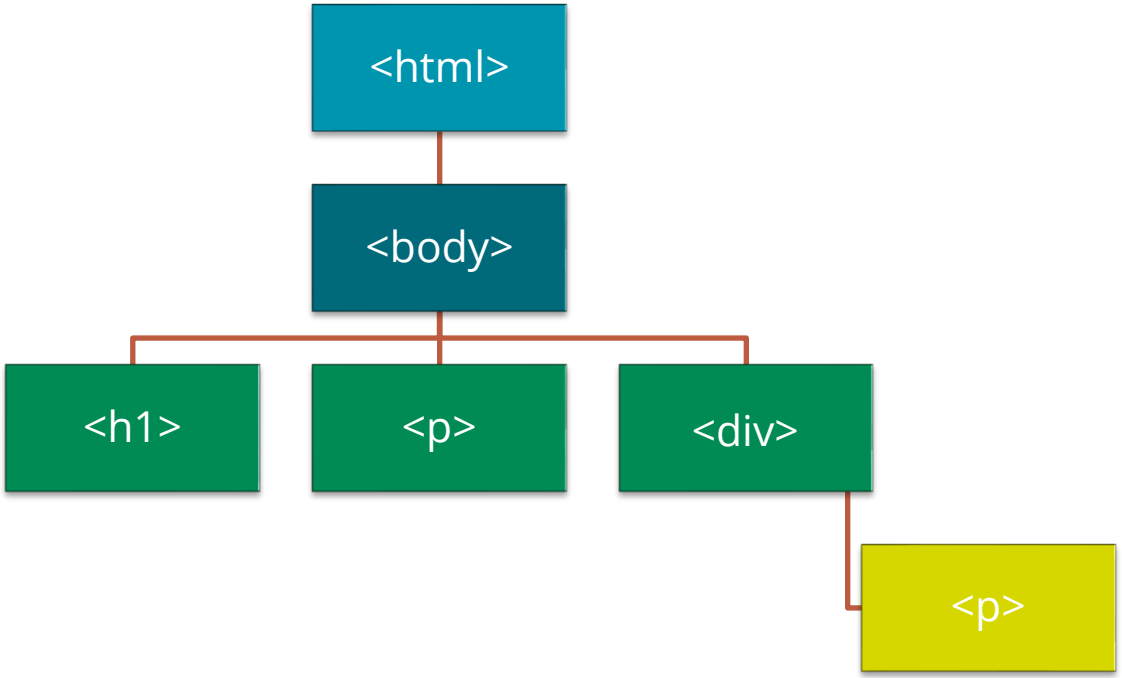
ESPECIFICIDAD

A	Número de estilos aplicados mediante un atributo **style** (si existe, siempre valdrá 1)
B	Número de apariciones de un **ID** en un selector
C	Número de apariciones de una **clase**, **pseudoclase** o **atributo** en el selector
D	Número de apariciones de un **elemento** o **pseudoelemento** en el selector

```
h1 /** A=0, B=0, C=0, D=1, peso 0001 */
h1 + p::first-line /** A=0, B=0, C=0, D=3, peso 0003 */
li > a[title*=“lemoncode”] > .selected /** A=0, B=0, C=2, D=2, peso 0022 */
#myID /** A=0, B=1, C=0, D=0, peso 0100 */
<a style=“color: red;”> /** A=1, B=0, C=0, D=0, peso 1000 */
```

Calculadora Especificidad Keegan

Concepto de herencia

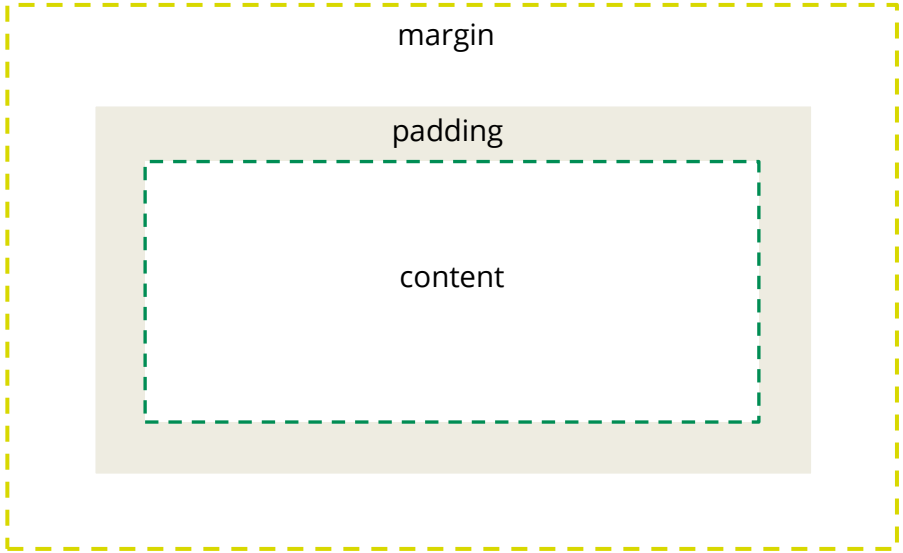


Control de herencia

inherit	Hereda el valor de la propiedad del elemento padre
initial	Establece el valor que tenía la propiedad inicialmente
unset	Hereda el valor de la propiedad del elemento padre, y en caso de no existir, su valor inicial

Modelo de caja

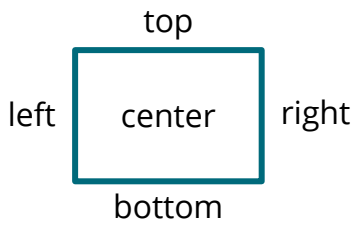
Cualquier elemento del documento tiene la estructura de una caja rectangular



EJEMPLO

Padding, margin y border

Zonas de un elemento



Atajo para márgenes y espaciado

[top, right, bottom, left] 1 valor = 2px
[top, bottom] [right, left] 2 valores = 2px 6px
[top] [right, left] [bottom] 3 valores = 2px 6px 4px
[top] [right] [bottom] [left] 4 valores = 2px 6px 6px 4px

```
ul li {
 padding-left: 5px;
 margin: 0 3px;
 border-bottom: 3px solid black;
}
```

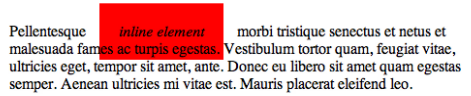
Propiedad display

La propiedad *display* controla el comportamiento de la caja tanto externo (cómo fluye en el documento) como interno (cómo se organizan los elementos hijos)

Comportamiento externo

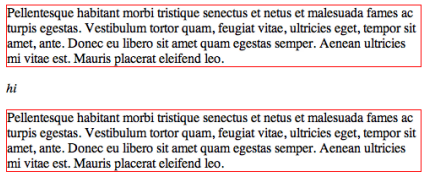
```
display: inline;
```

Por defecto en elementos como , , <b>...




```
display: block;
```

Por defecto en elementos como , <section>, <p>...




```
display: inline-block;
```

Combina los dos anteriores, se comporta como *inline* pero con *width* o *height*

Comportamiento interno

```
display: flex;
```

```
display: grid;
```

```
display: table;
```

y muchos más ...

EJEMPLO

Posicionamiento

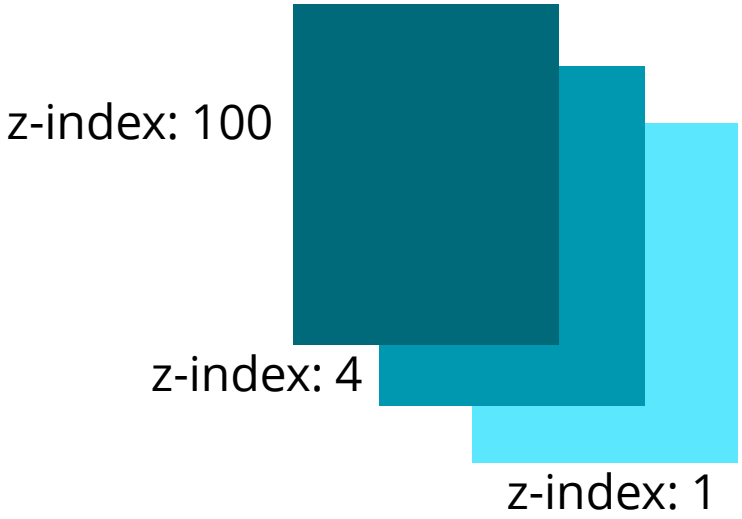
Para extraer un elemento del flujo normal del documento usamos la propiedad *position*

static	Se posiciona de acuerdo al flujo normal del documento (top, right, bottom, left y z-index no tienen efecto). Es el valor por defecto
relative	Se posiciona de acuerdo al flujo normal del documento y luego **es desplazado en relación a sí mismo**
absolute	Fuera del flujo normal y **su posición final es determinada en base al padre por los valores top, right, bottom y left**
fixed	Posicionamiento fijo. Igual que el absoluto pero no se mueve aunque hagamos scroll

EJEMPLO

Profundidad

Sirve para indicar el nivel de profundidad en la que está un elemento sobre los demás



Debe tener un position distinto de static

EJEMPLO

LEMONCODE

lemoncode.net

github.com/lemoncode

@lemoncoders

facebook.com/lemoncoders